Universidad Nacional-Sede Manizales

CH-MÁQUINA

MANUAL TÉCNICO

Alejandra Giraldo Quintero - 908526

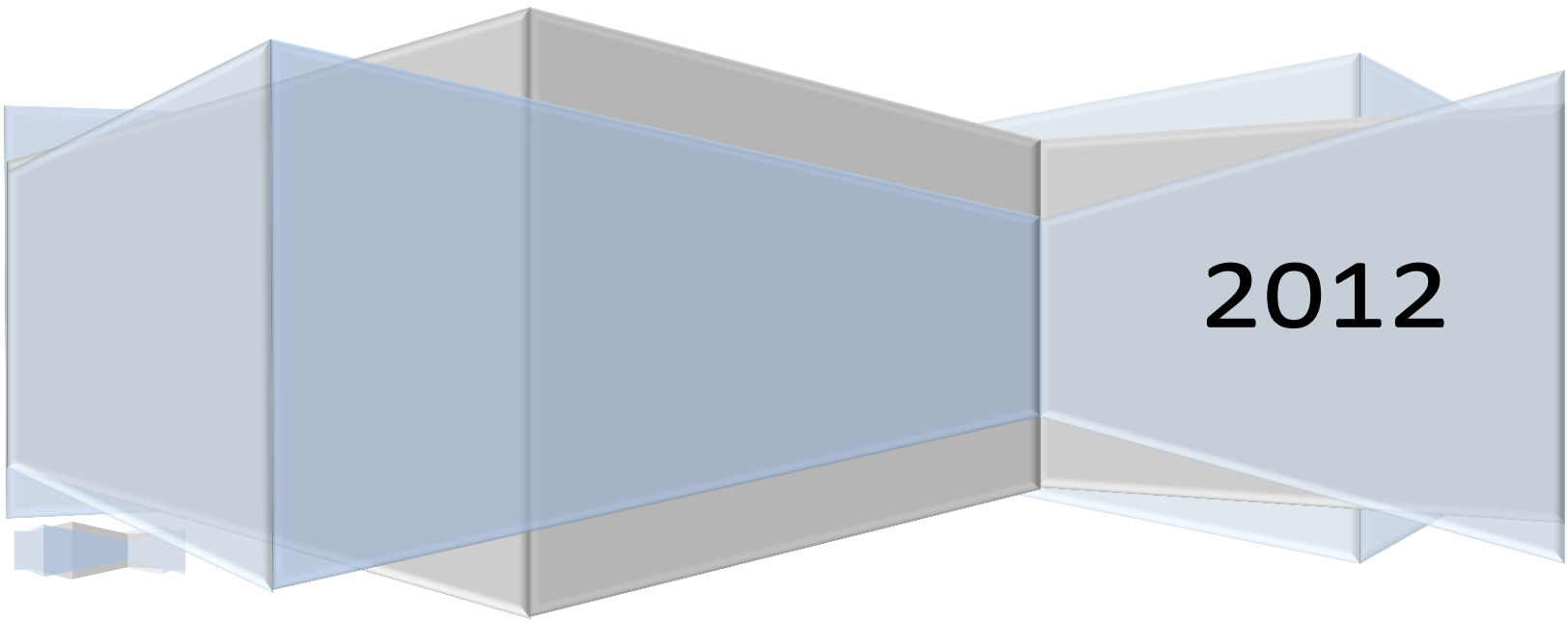

TABLA DE CONTENIDO

Introducción.....**2**
CH-Lenguaje.....**3**
Funcionalidades del CH-Computador.....**5**
Reglas del CH-Computador.....**6**
Especificaciones Técnicas.....**8**

INTRODUCCIÓN

El CH-Máquina, se refiere a un ambiente operativo que correrá sobre la Web, mediante este, se esta realizando una simulación gráfica de un computador ficticio de funcionamiento básico, al cual denominaremos CH-Computador.

Sobre el CH-Computador se podrán correr programas con extensión .ch, teniendo en cuenta de que estos deben estar escritos en un seudo-lenguaje denominado CH-Lenguaje, el cual esta basado en operaciones mínimas que puede realizar cualquier lenguaje de programación; mas adelante y con mas detalle se explicaran cada una de las instrucciones de este lenguaje para una mejor comprensión no solo de este, sino del CH-Computador como tal. Cabe resaltar que los programas .ch pueden cargarse desde cualquier directorio o si se desea estará disponible la opción de crearlos desde cero o editar alguno desde el CH-Computador.

Como bien debemos comprender, el manejo de la memoria en una computadora es de vital importancia, pues a partir de esta el sistema operativo podrá llevar a cabo una serie de operaciones, que basándose en diversos vectores de almacenamiento que integra el CH-Computador, permite responder a las exigencias que tiene este mismo; así en el espacio reservado de memoria, vamos a encontrar no solo el espacio para el Kernel (principal responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora (encargado de gestionar recursos a través de servicios de llamada al sistema))(Wikipedia), sino además para los programas .ch y para las variables y etiquetas que estos programas determinen dentro de sus instrucciones; y añadiendo a lo anterior una posición de memoria reservada para el acumulador (que permitirá registrar los valores de los cálculos, ofreciendo mayor facilidad para llevar a cabo cálculos interno).

Este simulador de ambiente operativo permitirá entonces un acercamiento profundo a la forma en la cual funciona un Sistema Operativo Real, respondiendo así a las exigencias y objeto del por que efectuamos este CH-Computador.

CH-LENGUAJE

El CH-Computador debe estar en la capacidad de leer un conjunto de programas en un seudo-lenguaje de maquina, para que así puedan cargarse en las posiciones de la memoria y pueda llevarse a cabo la ejecución de las instrucciones que contiene cada uno de ellos. Dado que las instrucciones que contienen los programas .ch, deben estar por lo menos sintácticamente correctas, a continuación se mostrara cuales son las instrucciones permitidas, con su descripción y sus parámetros, con el fin de que mas adelante se pueda entender no solo la verificación sintáctica que se lleva a cabo, sino además la lógica de funcionamiento.

Operación	Descripción	Sintaxis
cargue	Cárguese/copie en el acumulador el valor almacenado en la variable indicada por el operando.	**cargue**[nombreVariable]
almacene	Guarde/copie el valor que hay en el acumulador a la variable indicada por el operando.	**almacene**[nombreVariable]
vaya	Salte a la instrucción que corresponde a la etiqueta indicada por el operando y siga la ejecución a partir de allí.	**vaya**[nombreEtiqueta]
vayasi	Vaya a la instrucción que corresponde a la etiqueta indicada por el primer operando si el valor del acumulador es mayor a de cero, a la instrucción que corresponde a la etiqueta indicada por el segundo operando si el valor del acumulador es menor a cero o a la siguiente instrucción adyacente a la instrucción vayasi si el acumulador es cero y siga la ejecución a partir de allí.	**vayasi**[nombre1Etiqueta] [nombre2Etiqueta]
nueva	Crea una nueva variable cuyo nombre es el especificado en el primer operando, en el segundo operando definirá el tipo de variable (C Cadena/alfanumérico, I Entero, R Real/decimal), un tercer operando establecerá un valor de inicialización; a cada variable se le asignará automáticamente una posición en la memoria. Las variables deberán estar definidas antes de ser utilizadas. Las variables no inicializada tendrán por defecto el valor cero para reales y enteros y espacio para cadenas. El separador de decimales es el punto.	**nueva**[nombreVariable][tipoVariable][*ValorInicial* (Opcional)]
etiqueta	Crea una nueva etiqueta cuyo nombre es el especificado en el primer operando y a la cual le asignará automáticamente la posición indicada en el segundo operando (este será la posición relativa a la primera instrucción del programa). Las instrucciones que definen etiquetas podrán definirse en cualquier posición del programa, pero en todo caso antes de la instrucción retorne.	**etiqueta**[nombreEtiqueta] [líneaCorrespondiente]
lea	Lee por teclado el valor a ser asignado a la variable indicado por el operando.	**lea** [nombreVariable]
sume	Incremente el valor del acumulador en el valor indicado por la variable señalada por el operando.	**sume** [nombreVariable]
reste	Decremente el acumulador en el valor indicado por la variable que señala el operando.	**reste** [nombreVariable]
multiplique	Multiplique el valor del acumulador por el valor indicado por la variable señalada por el operando.	**multiplique**[nombreVariable]
divida	Divida el valor del acumulador por el valor indicado por la variable señalada por el operando. El divisor deberá ser una	**divida**[nombreVariable]

	cantidad diferente de cero.	
potencia	Eleve el acumulador a la potencia señalada por el operando(los exponentes deben ser valores enteros, positivos o negativos).	**potencia**[nombreVariable]
modulo	Obtenga el modulo al dividir el valor del acumulador por el valor indicado por la variable señalada por el operando.	**modulo**[nombreVariable]
concatene	Genere una cadena que una la cadena dada por el operando a la cadena que hay en el acumulador.	**concatene**[cadena]
elimine	Genere una subcadena que elimine cualquier aparición de los caracteres dados por el operando de la cadena que se encuentra en el acumulador.	**elimine**[caracteres]
extraiga	Genere una subcadena que extraiga los primeros caracteres (dados por el operando) de la cadena que se encuentra en el acumulador.	**extraiga**[numero]
muestre	Presente por pantalla el valor que hay en la variable por el operando, si el operando es acumulador muestre el valor del acumulador.	**muestre**[nombreVariable]
imprima	Lo m ismo que el anterior pero presentándolo en la impresora.	**imprima**[nombreVariable]
retorne	El programa termina; debe ser la última instrucción del programa y tiene como operando opcional un valor numérico de 0 a 255.	**retorne**[*valor*(opcional)]

Palabras Reservadas del CH-Lenguaje

{“cargue”, “almacene”, “vaya”, “vayasi”, “nueva”, “etiqueta”, “multiplique”, “lea”, “sume”, “reste”, “divida”, “potencia”, “modulo”, “concatene”, “elimine”, “extraiga”, “muestre”, “imprima”, “retorne”, “acumulador”}

Tipos de Variables

"I"→Entero

"C"→Carácter/Cadena (string)

"R"→Real/Decimal

Comentarios

Se denotan con Backslash → //

FUNCIONALIDADES DEL CH-Computador

El programa simula un procesador muy elemental y una memoria principal a través de un vector de hasta 99999 posiciones, las cuales pueden ser variadas al momento de iniciar el programa, se asume por defecto que el CH-Computador empieza con *100* posiciones de memoria para facilitar el proceso de pruebas. El programa debe estar en capacidad de leer un conjunto de programas en un seudo - lenguaje de máquina que llamaremos **CH-Lenguaje** y los cargara en las posiciones disponibles de la citada memoria, leerá una instrucción por cada línea de entrada.

Las primeras posiciones de la memoria estarán reservadas para el núcleo del sistema operativo (kernel), el tamaño de este deberá poderse ingresar al iniciar la corrida del simulador, en caso de que no se ingrese su tamaño, su valor por defecto es 79 . El mapa de memoria siempre estará visible, al igual que el valor del acumulador. Las posiciones de memoria que almacenen datos tendrán un identificativo así: CHAG_ _ _ (CHAG + tres dígitos numéricos).

El programa deberá realizar un chequeo de Sintaxis, produciendo un listado de errores si los hay, de lo contrario procederá a la carga definitiva del programa en memoria y quedará listo para ejecución del mismo bajo las reglas de corrida de múltiples programas.

REGLAS DEL CH-Computador

Se asume que el kernel ocupa las primeras posiciones de la memoria, su contenido no es importante y su tamaño se puede variar solo al iniciar el CH-Computador.

El programa utilizará un acumulador para registrar los valores de los cálculos y recibirá como nombre reservado “acumulador”.

Toda posición de memoria que se utilice para almacenar datos, deberá ser creada antes de ser usada y tendrá un nombre bajo las reglas ya establecidas.

Las instrucciones constan de 2 partes; el código de la operación y el(los) operando(s) dependiendo el tipo de instrucción.

La ejecución de los programas normalmente se hace de forma secuencial de instrucciones, la primera después la segunda, la tercera....etc., las instrucciones de transferencia de control son la forma de cambiar este orden de ejecución, obligando que el programa no siga en el orden secuencial predeterminado, sino que continué en la instrucción señalada por una etiqueta (es decir una instrucción que tiene asignado un nombre como referencia). Vaya y vaya si cumple esta función, la primera de forma incondicional y la segunda condicionada al valor del acumulador como se especifica en su definición.

La inicialización de posiciones de memoria se hará como instrucciones en las cuales se crean las variables y se les asigna valor, como se explico en la operación Nueva.

Se podrán realizar operaciones entre valores enteros y reales, los resultados intermedios se manejaran como reales y el resultado final obedecerá al tipo de variable que almacena el resultado.

La protección de memoria se hará por registro base y registro límite, esto es, cada programa empieza en una posición de memoria (registro base) y termina en otra posición de memoria denominada (registro limite) con base en las cuales se evitará la violación de las normas básicas de ejecución, también debe tenerse claro que los programas tendrán área de código y área de datos.

El programa podrá ejecutarse en modalidad normal (corrida continua) o paso a paso(instrucción por instrucción), en todo caso se podrá visualizar la instrucción que se esté ejecutando en cada momento y el respectivo valor del acumulador.

Los CH-Programas serán almacenados previamente en archivos con extensión .ch en cualquier carpeta de algún medio de almacenamiento, de allí podrán ser cargados al señalarlos de la lista. Además el CH-Computador provee un editor de programas .ch, en el cual se puedan crear CH-Programas y/o modificarlos, este editor hará el chequeo de sintaxis básico indicándole al usuario las instrucciones con errores.

Se podrán cargar y correr varios CH-Programas hasta agotar la memoria disponible, para la corrida de los CH-Programas se utiliza una cola circular, la cual será visitada con base al orden de llegada/cola simple (primero en entrar primero en ser atendido-FCFS).

El sistema indica por medio de alguna convención si está trabajando en modo usuario o modo kernel, mostrando la acción de cambio de contexto.

ESPECIFICACIONES TECNICAS

Para la elaboración de este CH-Computador se utilizo lo siguiente:

Lenguaje de Programación: C#, ASP.NET con el Framework 4.0 (Visual Studio 2010)

Ahora en cuanto a la funcionalidad:

Se creo un proyecto denominado CH-Maquina, dentro de este se crearon 5 clases así:

archivo.cs → Esta clase es la que se encarga de abrir los programas .ch, para luego leer línea tras línea de estos agregándolos así en un arreglo de tipo string, es decir por cada posición del arreglo estarán una a una las líneas del programa correspondiente. Cuando ya se hayan efectuado las anteriores actividades, se cerrara el archivo. Hay que tener en cuenta que paralelo a esto, se van contando las líneas que tiene el programa.

verificación.cs→Lo que hará básicamente esta clase es cumplir con el requisito que se exige de que se debe realizar un chequeo de Sintaxis, produciendo un listado de errores si los hay, para que así pueda cargarse definitivamente el programa en memoria y quede listo para la ejecución del mismo.

programa.cs→Se encarga de la administración de las variables en general que maneja un programa .ch, así como de la definición de los registros bases y limite de los que se hablaba anteriormente.

memoria.cs→Esta clase es la encargada de todo lo referente a la memoria del CH-Computador, aquí se definirá no solo su tamaño, sino a su vez los métodos necesarios para ajustar el espacio del kernel, el espacio de los programas y de las variables. Además del reinicio de la memoria y la carga de elementos en esta.

reglasOperación→Es quien interpreta las instrucciones de los CH-Programas para poder ejecutarlos. Indicando todo lo que realmente debe hacer cada línea de instrucción que este dentro de los programas .ch.

Ahora es de vital importancia para el usuario el orden en el cual fueron escritas las clases anteriores, puesto que de esta forma es que se concibe la funcionalidad como tal del CH-Computador. Debido a que:

1. archivo.cs, el archivo ya abierto o traído desde el editor que facilita el CH-Computador pasa a:

2. verificacion.cs, para que esta clase haga una revisión sintáctica del contenido del programa.ch, si no se encuentra ningún error entonces:

3. programa.cs, crea una instancia de programa, el cual es llevado a:

4. memoria.cs, la memoria y así estando ya en memoria pueda:

5. reglasOperacion.cs, ejecutar en la modalidad que indica cada una de las instrucciones contenidas en los programas cargados que ya están en la memoria.

CONTINUA CON UNIFICACION DE LA PARTE WEB.